FORGERY ON STATELESS CMCC WITH A SINGLE QUERY

Guy Barwell

University of Bristol

Abstract. We present attacks against CMCC that invalidate the claimed security of integrity protection and misuse resistance. We exploit the fact zero-padding is used on both the message and authenticated data and demonstrate how one may generate a forgery with a single call to the encryption oracle. From this we calculate the ciphertext of the chosen message, yielding a forgery and so breaking INT-CTXT. In the nonce-reuse setting, existence of a forgery leads directly to a 2-query distinguisher.

1. Description of Stateless CMCC

CBC-Mac-Counter-CBC[5] (henceforth CMCC) is a CAESAR[1] submission, and comes in both stateful and stateless forms. In this note we consider the stateless version, which is the recommended configuration, and demonstrate a weakness in the mode of operation itself. As such, our attacks holds across all stateless parameter sets, irrespective of the choice of primitives.

1.1. Notation.

Notation. Following the original paper, let B be the blocksize in bits,τ the number of authenticity bits and N a Public Message Number which must be a nonce, with recommended values $(B, \tau, |N|) = (128, 64, 32)$. No secret message number is used in the stateless version. Let $\oplus$ and $||$ denote respectively the xor and concatenation of two strings. Constant bytes are provided in hexadecimal and typeset in typewriter font (eg `0xB6`). Finally, 0^α is the string of α zero bits.

Whilst all lengths will be given in bits, as per submission requirements[1] they shall all be exact number of bytes, and thus multiples of 8.

Where appropriate, $\mathcal{E}_K$ represents an encryption oracle, whilst $\mathbf{CMCC}_K$ is the CMCC encryption function under key K.

1.2. Components.

Components. For clarity of notation, we will describe CMCC in terms of the following well known components, each instantiated with an appropriate block-cipher (for which the recommendation is AES):

- Let $\mathbf{Pad}_b(M)$ be the function that returns bitstring M padded up to b bits by appending sufficiently many zero bits.
- Let $\mathbf{MSB}_b(M)$ be the Most Significant Bits function, returning bitstring M truncated to the first b bits.
- Let $\mathbf{E}_k(m)$ be the encryption of a single block m with key k using the block cipher.
- Let $\mathbf{CBC}_k^N(M)$ be the cipher block chaining mode[3] encryption of message M under key k and initial value N.
- Let $\mathbf{MAC}_k^N(M)$ be an unforgable MAC on message M under key k and with initial value N. The recommended instantiation is AES-CMAC[4].

Algorithm 1 Stateless CMCC Encryption if $2|A| + |M| + \tau \le 2B$

```
(1.1)        $\mathbf{CMCC}(N, A, M)$
(1.2)        $\bar{K}, L_1, L_2, \bar{L_2}, L_3 \leftarrow \mathbf{KeyExpansion}(K)$
(1.3)            $W \leftarrow \mathbf{E}_{\bar{K}}\left((\texttt{0xB6})^{B-|N|}||N\right)$
(1.4)            $P_1||P_2 \leftarrow M||0^\tau$        (st. $|P_1| \in \{|P_2|, |P_2| - 8\}$)
(1.5)            $R_0 \leftarrow \mathbf{Pad}_B(P_1)$
(1.6)            $R_1 \leftarrow \mathbf{CBC}^W_{L_3}(R_0)$
(1.7)            $X \leftarrow \mathbf{MSB}_{|P_2|}(R_1) \oplus P_2$
(1.8)            $T_1 \leftarrow \mathbf{Pad}_B(X||A)$
(1.9)            $T_2 \leftarrow \mathbf{E}_{L_2}(T_1)$
(1.10)           $X_2 \leftarrow \mathbf{MSB}_{|P_1|}(T_2) \oplus P_1$
(1.11)           $S_0 \leftarrow \mathbf{Pad}_B(X_2)$
(1.12)           $S_1 \leftarrow \mathbf{CBC}^W_{L_1}(S_0)$
(1.13)           $X_1 \leftarrow \mathbf{MSB}_{|P_2|}(S_1) \oplus X$
(1.14)       $\mathbf{Return}\, X_1, X_2$
```

- Let $\mathbf{CTR}^N_k(M)$ be the counter mode encryption[2] of M under key k with initial counter value N.

The internal mechanism $\mathbf{KeyExpansion}(K)$ is instantiated by the scheme as simply splitting a key K several block lengths long into separate keys.

1.3. **CMCC Encryption.** We split the two possible cases of the algorithm into two separate listings. In Algorithm 1 we provide the algorithm used to encrypt if the message,associated data and tag lengths satisfy $2 \cdot |A| + |M| + \tau \le 2B$, and Algorithm 2 provides the code used otherwise.

The decryption algorithm on input (K, N, A, X_1, X_2) uniquely recovers $P_1||P_2$ from 1.4 (2.4 in the long version) and if this string parses as $M||0^\tau$ returns that M. Otherwise (if the string does not end with 0^τ) the string fails authentication and decryption returns $\perp$. For a full listing of the algorithm, see the original paper [5].

2. Attacks on stateless CMCC

We now describe efficient forgeries against the two variants.

2.1. **Forgery on stateless CMCC for short messages.** Let (N, A, M) be an encryption input triple such that $2(|A| + 8) + |M| + \tau \le 2B$ (for example, it would suffice to have an empty message and half a block of associated data). We produce a forgery by appending a zero byte to the associated data and submitting the same ciphertext.

Consider the difference between the internal variables of the routine when encrypting (N, A, M) and $(N, A||\texttt{0x00}, M)$. The only place the input variables differ is on line 1.8, where in the former $T_1 \leftarrow \mathbf{Pad}_B(X||A)$, and in the latter $T_1 \leftarrow \mathbf{Pad}_B(X||A||\texttt{0x00})$. However, since the padding is with zero bytes and up to the same length, in each case T_1 takes the same value. Thus for the whole

Algorithm 2 Stateless CMCC Encryption if $2|A| + |M| + \tau > 2B$

```
(2.1)      $\mathbf{Encrypt}(N, A, M)$
(2.2)  $\bar{K}, L_1, L_2, \bar{L_2}, L_3 \leftarrow \mathbf{KeyExpansion}(K)$
(2.3)          $W \leftarrow E_{\bar{K}}\left((\texttt{0xB6})^{B-|N|}||N\right)$
(2.4)          $P_1||P_2 \leftarrow M||0^\tau$          $(\text{st } |P_1| \in \{|P_2|, |P_2| - 8\})$
(2.5)          $R_0 \leftarrow \mathbf{Pad}_\beta(P_1)$          $(\beta = \min\{\beta \in B\mathbb{N} \colon \beta \geq |P_2|\})$
(2.6)          $R_1 \leftarrow \mathbf{CBC}^W_{L_3}(R_0)$
(2.7)          $X \leftarrow \mathbf{MSB}_{|P_2|}(R_1) \oplus P_2$
(2.8)          $V \leftarrow \mathbf{MAC}^W_{L_2}(X||A)$
(2.9)          $X_2 \leftarrow \mathbf{CTR}^V_{L_2}(P_1)$
(2.10)         $S_0 \leftarrow \mathbf{Pad}_\beta(X_2)$
(2.11)         $S_1 \leftarrow \mathbf{CBC}^W_{L_1}(S_0)$
(2.12)         $X_1 \leftarrow \mathbf{MSB}_{|P_2|}(S_1) \oplus X$
(2.13)     $\mathbf{Return}\, X_1, X_2$
```

of the calculation the internal variables are the same in each case. Therefore, by requesting the encryption $(X_1, X_2) = \mathcal{E}_K(N, A||\texttt{0x00}, M)$ from the encryption oracle, we can can produce the forgery on the associated data, since we know that $\mathbf{CMCC}_K(N, A, M) = (N, A, X_1X_2)$.

2.2. **Related ciphertexts from related messsages in stateless CMCC for Longer messages.** The key observation is that the method used to parse input on line 2.5 and apply padding on line 2.7 allow us to create two messages $M, \tilde{M}$ such that the majority of the internal variables take the same values whether calculating $\mathbf{CMCC}_K(N, A, M)$ or $\mathbf{CMCC}_K(N, A, \tilde{M})$. As a result, the ciphertext from one can be used to form the ciphertext of the other by modifying just the final byte.

Lemma 1. *Let (N, A, M) be a valid nonce, associated-data, message triple with $|M| = 2iB + 8 - \tau$ for some $i \in \mathbb{N}$ and $2|A| + |M| + \tau > 2B$. Write $M = M_1||M_2$ with $|M_1| = iB$ and $|M_2| = iB + 8 - \tau$, and set $\tilde{M} = M_1||\texttt{0x00}||M_2$. Then, if $(X_1, X_2) = \mathbf{CMCC}_K(N, A, M)$ and $(\tilde{X_1}, \tilde{X_2}) = \mathbf{CMCC}_K(N, A, \tilde{M})$, we have that $\mathbf{MSB}_{iB}(X_1) = \mathbf{MSB}_{iB}(\tilde{X_1})$, and $X_2 = \mathbf{MSB}_{iB}(\tilde{X_2})$.*

Proof. For clarity's sake, we let the internal variables from the encryption of M use the same notation as in Algorithm 2, and their counterparts during the second encryption shall use the same variable names, marked with a tilde. A direct comparison of the two encryptions is provided in Appendix A.

The first line on which the internal variables will differ is 2.4, where $(P_1, P_2) = (M_1, M_2||0^\tau)$, as opposed to $(\tilde{P_1}, \tilde{P_2}) = (M_1||\texttt{0x00}, M_2||0^\tau)$. In particular, we have that $\tilde{P_2} = P_2$ and $\tilde{P_1} = P_1||\texttt{0x00}$. $|P_1| = iB$, whilst the other three are of length $iB + 8$ bits. Therefore, on line 2.5 both P_1 and $\tilde{P_1}$ are zero padded up to the next block boundary, meaning $\tilde{R_0} = R_0 = M_1||0^B$. Thus the difference between P_1 and $\tilde{P_1}$ is hidden by the padding.

Results:	*Previous* [5]			New (short)			New (long)		
$\tau =$	64	32	8	64	32	8	64	32	8
Confidentiality of plaintext	128	128	128						
integrity for plaintext	64	32	16				7	7	7
integrity for Assoc. data	64	32	16	0	0	0			
integrity for PMN	64	32	16						

TABLE 1. Comparison of integrity goals for stateless AES-CMCC v1 with t byte tag

This directly leads to the internal variables across lines 2.6,2.7 and 2.8 being equal, meaning $(R_1, X, V) = (\tilde{R_1}, \tilde{X}, \tilde{V})$. Therefore, the streams generated by the counter on line 2.9 are also equal, and since $\tilde{P_1} = P_1||\texttt{0x00}$ we have $\tilde{X_2} = X_2||\chi$ for some unknown byte χ.

Now, line 2.10 again zero pads up to the next block boundary, and so $S_0 = X_2||0^B$, whilst $\tilde{S_0} = X_2||\chi||0^{B-8}$. Notice that, because P_1 was a multiple of the block length, the only difference between these values occurs in the final block. This is important since on line 2.11 we now encrypt S_0 and $\tilde{S_0}$ in CBC mode, which means that S_1 and $\tilde{S_1}$are equal on all but the final block. Finally, this means that X_1 and $\tilde{X_1}$ are equal on all but the final block. Since $|X_1| = |P_2| = iB + 8$, any difference between X_1 and $\tilde{X_1}$ occurs within the final8 bits. □

2.3. **Forgery on stateless CMCC for longer messages.** To produce a forgery in the INT-CTXT game, we pick an associated-data/message pair that satisfies the requirements of Lemma 1. For example: $(A, M) = (0^B, 0^{4B+8-\tau})$. Having constructed $\tilde{M}$ as above, we query the encryption oracle for $(\tilde{X_1}, \tilde{X_2}) = \mathcal{E}_K(N, A, \tilde{M})$ for some valid nonce. Let $X_2 = \mathbf{MSB}_{iB}(\tilde{X_2})$, and set $\tilde{X_1} = \mathbf{MSB}_{iB}(\tilde{X_1})$. From the result above, we know that the valid encryption of $\mathbf{CMCC}_K(N, A, M) = (X_1||\delta, X_2)$ for some byte δ. Thus to construct a forgery we simply guess all 2^8 possible values of δ, and query each one to the decryption oracle. One of these will be a valid ciphertext, and thus a valid forgery.

Therefore, with one query of the encryption oracle and at most 2^8 queries of the decryption oracle (expected number of queries 2^7) a forgery can be produced for stateless CMCC for longer messages.

3. Comparison with security claims of CBCC

3.1. **Integrity.** Table 2.1 of [5] provides integrity goals for the scheme, and is reproduced in the first three columns of Table 1. The next triplet of columns refer to the strength of the protection offered when a very short message is used, as investigated in Section 2.1, and the final set to those investigated in Section 2.3. The new results are significantly lower than the claimed security level.

3.2. **Robustness: Extending the forgery to an distinguisher against MRAE.** Section 2.1 of the CMCC submission document states that robustness against Public-Message-Number abuse (ie nonce reuse) should be optimal. However, since we are able to calculate a valid encryption for a known message without querying the oracle on it, we can construct a distinguisher against the misuse resistance claim. First, we follow the method given in this note to find the encryption for a message M we have not queried the oracle on. Then, after querying the oracle for $C = \mathcal{E}_K(N, A, M)$, we can compare C with our forgery, and thus distinguish a CMCC-Encrypt oracle from a Random Oracle with overwhelming probability.

Similarly, Section 2.2 states that it should not be possible to modify a valid ciphertext to cause a predictable change in the plaintext, something we demonstrate is possible.

4. Conclusion

We have presented attacks that demonstrate the integrity guarantees of CMCC are not as high as originally conjectured, and that some of the security claims of CMCC are therefore not met. This attack is due to the use of zero-padding, and it might be possible to resolve the weakness using a more robust padding scheme, even if the validity of paddings is not checked.

5. Acknowledgements

The author is grateful for helpful discussions and comments on a previous draft provided by Martijn Stam and Gaven Watson. Thanks also to Daniel Bernstein for coordinating the CAESAR competition and hosting the submissions.

This work was conducted whilst Guy Barwell a PhD student in the University of Bristol Cryptography-Security group. He was supported by an EPSRC grant.

References

[1] Daniel Bernstein. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. http://competitions.cr.yp.to/caesar.html, 2013. 1

[2] M Dworkin. Recommendation for Block Cipher Modes of Operations. *NIST SP 800-38A*, December 2001. 2

[3] FIPS. 81: DES Modes of Operation. *Issued December*, 2:63, 1980. 1

[4] JH Song, R Poovendran, J Lee, and T Iwata. RFC 4493: The AES-CMAC algorithm. Technical report, Technical report, Corporation for National Research Initiatives, Internet Engineering Task Force, Network Working Group, 2006. 1

[5] Jonathan Trostle. AES-CMCC: v1. Submitted to CAESAR competition, March 2014. 1, 2, 4

Appendix A. comparison between encryption $(N, A, M_1||M_2)$ and $(N, A, M_1||\texttt{0x00}||M_2)$

(2.1)	$\mathbf{Encrypt}(N, A, M_1\|\|M_2)$	$\mathbf{Encrypt}(N, A, M_1\|\|\texttt{0x00}\|\|M_2)$
(2.2)	$L_1, L_2, L_3 \leftarrow \mathbf{KeyExpansion}(K)$	$L_1, L_2, L_3 \leftarrow \mathbf{KeyExpansion}(K)$
	$\bar{K}, \bar{L}_2 \leftarrow \mathbf{KeyExpansion}(K)$	$\bar{K}, \bar{L}_2 \leftarrow \mathbf{KeyExpansion}(K)$
(2.3)	$W \leftarrow E_{\bar{K}}\left((\texttt{0xB6})^{16-\|N\|}\|\|N\right)$	$W \leftarrow E_{\bar{K}}\left((\texttt{0xB6})^{16-\|N\|}\|\|N\right)$
(2.4)	$P_1\|\|P_2 \leftarrow M_1\|\|M_2\|\|0^\tau$	$\tilde{P_1}\|\|\tilde{P_2} \leftarrow M_1\|\|\texttt{0x00}\|\|M_2\|\|0^\tau$
	$//P_1 = M_1, P_2 = M_2\|\|0^\tau$	$//\tilde{P_1} = P_1\|\|\texttt{0x00}, \tilde{P_2} = P_2$
	$//\|P_1\| = iB; \|P_2\| = iB + 8$	$//\|\tilde{P_1}\| = \|\tilde{P_2}\| = iB + 8$
(2.5)	$R_0 \leftarrow \mathbf{Pad}_{(i+1)B}(P_1)$	$R_0 \leftarrow \mathbf{Pad}_{(i+1)B}(P_1\|\|\texttt{0x00})$
(2.6)	$R_1 \leftarrow \mathbf{CBC}^W_{L_3}(R_0)$	$R_1 \leftarrow \mathbf{CBC}^W_{L_3}(R_0)$
(2.7)	$X \leftarrow \mathbf{MSB}_{\|P_2\|}(R_1) \oplus P_2$	$X \leftarrow \mathbf{MSB}_{\|P_2\|}(R_1) \oplus P_2$
(2.8)	$V \leftarrow \mathbf{MAC}^W_{L_2}(X\|\|A)$	$V \leftarrow \mathbf{MAC}^W_{L_2}(X\|\|A)$
(2.9)	$X_2 \leftarrow \mathbf{CTR}^V_{\bar{L}_2}(P_1)$	$\tilde{X_2} \leftarrow \mathbf{CTR}^V_{\bar{L}_2}(P_1\|\|\texttt{0x00})$
	$//\|X_2\| = \|P_1\| = iB$	$//\tilde{X_2} = X_2\|\|\chi$ (for some byte χ)
(2.10)	$S_0 \leftarrow \mathbf{Pad}_{(i+1)B}(X_2)$	$\tilde{S_0} \leftarrow \mathbf{Pad}_{(i+1)B}(X_2\|\|\chi)$
	$//S_0 = X_2\|\|0^B$	$//\tilde{S_0} = X_2\|\|\chi\|\|0^{B-8}$
(2.11)	$S_1 \leftarrow \mathbf{CBC}^W_{L_1}(X_2\|\|0^B)$	$\tilde{S_1} \leftarrow \mathbf{CBC}^W_{L_1}(X_2\|\|\chi\|\|0^{B-8})$
		$//\mathbf{MSB}_{iB}(\tilde{S_1}) = \mathbf{MSB}_{iB}(S_1)$
(2.12)	$X_1 \leftarrow \mathbf{MSB}_{iB+8}(S_1) \oplus X$	$\tilde{X_1} \leftarrow \mathbf{MSB}_{iB+8}(\tilde{S_1}) \oplus X$
		$//\mathbf{MSB}_{iB}(\tilde{X_1}) = \mathbf{MSB}_{iB}(X_1)$
		//ie: they only differ on final byte
(2.13)	**Return** X_1, X_2	**Return** $\tilde{X_1}, \tilde{X_2}$